Assessing the Effectiveness of Acceleration Methods for Deterministic Neutron Transport Solvers

J. S. Rehak,[a,*] R. N. Slaybaugh,[a]

[a] *Department of Nuclear Engineering, University of California, Berkeley*
jsrehak@berkeley.edu

INTRODUCTION

The neutron transport problem is often computed by various iterative methods. These iterative methods suffer from poor convergence properties in many applications of practical interest for both dominant eigenvalue criticality and fixed source problems. This has necessitated the development of a broad field of acceleration methods to improve their convergence properties. The two primary measures of effectiveness of acceleration methods are reduction in computational work, typically measured through the number of inversions of the transport operator, or "sweeps", and reduction in runtime. Neither measure, however, tells us *how* the acceleration is happening or whether the improvement is occurring for the reasons we think it is. If we had better information about *why* a code is performing better in practice, we would be better able to confirm our hypotheses about new methods and ensure they will work as expected for problems of interest.

In this work we present the design goals and status of a new code, Bay Area Radiation Transport (BART) [1], which is designed to assess the effectiveness of acceleration methods. BART is a highly instrumented code, collecting granular data about error modes, error reduction, and convergence behavior. BART is set up so the addition of new measures and collection of additional data is easy to add such that a developer can easily add any instrumentation they desire. Further, BART is written in a highly modular way so adding new methods requires minimal change to code – enhancing the ability to make a direct comparison between methods.

To illustrate how and why BART may be a powerful research tool, we will provide more specific explanation using legacy methods in the field as examples. These methods have been widely used and are fairly straightforward to understand. The layout of this paper is as follows: the source and power iteration methods will be described, followed by a discussion of assessing the effectiveness of acceleration methods, and then the design goals and description of the new code are given.

ITERATIVE METHODS

The formulations and descriptions in this section are derived from Lewis and Miller [1]. The time-independent linear Boltzmann transport equation is

$$\begin{aligned}\hat{\Omega}\cdot\nabla\psi\left(\vec{r},\hat{\Omega},E\right)+\Sigma_t\left(\vec{r},E\right)\psi\left(\vec{r},\hat{\Omega},E\right)\\ =\int_0^\infty dE'\int_{4\pi}d\hat{\Omega}'\,\Sigma_s\left(\vec{r},\hat{\Omega}'\rightarrow\hat{\Omega},E'\rightarrow E\right)\psi\left(\vec{r},\hat{\Omega}',E'\right)\\ +q(\vec{r},\hat{\Omega},E)\,,\end{aligned} \tag{1}$$

where the terms are defined as follows:

$\vec{r}$ = position in a spatial domain of interest,
E = neutron energy,
$\hat{\Omega}$ = normalized angle of neutron travel, such that $\left|\hat{\Omega}\right| = 1$,
ψ = angular flux,
Σ_t = macroscopic total cross-section,
Σ_s = double differential macroscopic scattering cross-section,
q = non-scattering neutron source.

Eq. (1) obeys the boundary condition

$$\psi(\vec{r},\hat{\Omega},E)=\Gamma(\vec{r},\hat{\Omega},E),\quad\forall\vec{r}\in\partial\mathcal{D},\quad\forall\hat{\Omega}\cdot\hat{n}<0\,, \tag{2}$$

where Γ is a given function and $\hat{n}$ is the outward normal on the boundary.

We derive the multigroup S_N equations by discretizing energy and angle. The energy domain is discretized into $G+1$ groups on the range $[0, G]$, where zero is the highest energy and G the lowest, and a Petrov-Galerkin scheme is applied. In angle, we replace the angular integral with an $(N+1)$-point numerical quadrature scheme that also serves as a collocation scheme in angle. By using a quadrature scheme that exactly integrates the spherical harmonics, we can expand the scattering cross-section in the Legendre polynomials and express the angular flux in terms of flux moments. In matrix form, this is

$$\vec{\Phi}=\mathbf{D}\vec{\Psi}\,, \tag{3}$$

where the vector $\vec{\Psi}$ is the angular flux at the discrete collocation points for each group and $\mathbf{D}$ converts this vector to spherical-harmonic moments $\vec{\Phi}$.

The multigroup S_N equations in operator form are then

$$\mathbf{L}\vec{\Psi}=\mathbf{MS}\vec{\Phi}+\vec{q} \tag{4}$$

Where $\mathbf{L}$ is the transport operator, $\mathbf{M}$ is the moment-to-discrete operator, and $\mathbf{S}$ is the scattering moments. The large number of dimensions in the solution space of this equation often makes solving the problem directly impractical. Therefore, to solve real problems that have useful discretizations with accuracy, we rely on fast iterative methods.

The source iteration (SI) scheme [2] applied to Eq. (4) gives

$$\mathbf{L}\vec{\Psi}_{(\ell+1)}=\mathbf{MS}\vec{\Phi}_{(\ell)}+\vec{q}\,, \tag{5}$$

where $\ell\geq 0$ is the iteration number and $\vec{\Phi}_{(0)}$ is a specified initial guess. In practice, this is executed by calculating the flux moments from the solution of the previous iteration and solving for the new angular flux until converged. The angular flux for iteration $\ell+1$ represents the angular flux for particles that have scattered at most ℓ times. This leads to rapid

[1] `https://github.com/SlaybaughLab/BART`

convergence for systems with little scattering, but arbitrarily slow convergence for systems with a large amount of scattering [2]. This has the largest impact in solving problems with materials designed for a high amount of scattering but a very small amount of absorption, such as moderators. Therefore, accelerating the source-iteration process for these situations is of value.

In a multiplying media, we include a fission term that is a function of the flux moments and the fission eigenvalue k. This changes Eq. (4) to

$$\mathbf{L}\vec{\Psi} = \mathbf{M}\left[\mathbf{S}\vec{\Phi} + \frac{1}{k}\mathbf{F}\vec{\Phi}\right] + \vec{q}\,, \tag{6}$$

where k is the eigenvalue, $\mathbf{F}$ is the discretized fission operator,

$$\mathbf{F} = \mathcal{F}\psi\left(\vec{r}, \hat{\Omega}, E\right) = \chi(E)\int_0^\infty dE'\,\nu(E')\Sigma_f(\vec{r}, E')\int_{4\pi} d\hat{\Omega}\,\psi(\vec{r}, \hat{\Omega}, E')\,,$$

$\chi(E)$ is the fission spectrum, and $\nu(E)$ is the mean number of neutrons produced by a fission caused by a neutron of energy E.

In practice, this is solved by fixing the scattering source and solving for the fission source. This process is power iteration (PI). The power iteration form of Eq. (6) is defined

$$\mathbf{L}\vec{\Psi}_{(\ell+1)} = \mathbf{M}\left[\mathbf{S}\vec{\Phi}_{(0)} + \frac{1}{k}\mathbf{F}\vec{\Phi}_{(\ell)}\right] + \vec{q}\,, \tag{7}$$

where $\vec{\Phi}_{(0)}$ is a fixed flux moment. The scattering source flux moment is usually solved by a source-iteration scheme that holds the fission source constant. This forms an inner (source iteration) and outer (power iteration) iteration scheme that will converge both the scattering and fission sources.

Power iteration will converge on the largest absolute value eigenvalue, which is the fundamental mode and is therefore of practical interest. The convergence rate of PI is proportional to the ratio of the second-largest absolute value eigenvalue to the first, called the dominance ratio. If the dominance ratio is close to one, PI will converge arbitrarily slowly. The dominance ratio can be close to unity [3] in common reactor problems.

As discussed in this section, solving common problems of interest using source or power iteration may result in arbitrarily slow convergence. Overcoming this issue is the main focus of a broad family of *acceleration methods*.

ASSESSING ACCELERATION METHODS

The primary goal of acceleration methods is to reduce the total amount of computational work required for an iterative method to converge. Adams and Larson [2] provide an in-depth discussion of the history and development of acceleration schemes. Importantly, they also discuss the mathematical basis for why an iterative scheme will ultimately converge to the solution of the problem. For the purpose of this discussion, we shall assume that the given schemes will converge to a true solution, even if it requires an arbitrary amount of work. To achieve this convergence, the error between our initial guess and the true solution must be reduced to some tolerance. At a given iteration ℓ in an iterative scheme, the remaining error is given as

$$\vec{e}_{(\ell)} = \vec{\Psi}^* - \vec{\Psi}_{(\ell)} \tag{8}$$

where $\vec{\Psi}^*$ is the true solution. Convergence may then be defined by taking a norm of the error and comparing it to a set threshold value. In general, we do not know the true solution *a priori*, so calculating the true error in each iteration is impossible. Instead, we use the difference between our current solution and the previous solution to assess the magnitude of the true error. Adams and Larson [2] provide a discussion of the well-known issues with this approach.

Each iteration of an iterative method removes a portion of the remaining error via some amount of computational work. At convergence, the total error has been reduced to meet some convergence criteria. If this was performed in N iterations, this process requires total work,

$$W = \sum_{\ell=1}^{N} w_{(\ell)} \tag{9}$$

where $w_{(\ell)}$ is the computational work required for iteration ℓ.

If an acceleration method is introduced, the iterative process is modified but the convergence tolerance is held the same. The error reduction for each iteration should be different and the number of iterations required for convergence may be different as a result. The amount of work for each iteration is likely to be different, but the same convergence criteria must be met.

Changes to the iterative process have accelerated the process if less work was needed to achieve the error reduction required for convergence:

$$\sum_{\ell=1}^{N} w_{(\ell)} > \sum_{\ell=1}^{N'} w'_{(\ell)}\,. \tag{10}$$

The total error that needs to be removed for a problem to converge is independent of the specific iterative method used. If a modified method converges, it has removed approximately the same amount of error, leaving us to measure work to determine if the process has been accelerated. We note that work is a more consistent measure than runtime due to variations in computer hardware and auxiliary use at the time of measure. A method that consistently reduces work should consistently reduce runtime. The measurement of work is the topic of the following section.

Measuring work

There are many types of iterative methods for solving the neutron transport equation that divide computational work in different ways. When we examine Eqs. (5) and (7), we find that solving each iteration requires an inversion of the very large transport operator matrix $\mathbf{L}$. This may be done explicitly or implicitly. The more common implicit inversion may be performed by using a *transport sweep*, which solves

cells or groups of cell sequentially until the entire domain is solved. In either case, this inversion dominates the computational work in each iteration of the solve. We can therefore simplify the calculation of work in Eq. (9) as

$$W = \sum_{\ell=0}^{N} w_{(\ell)} \approx \sum_{\ell=0}^{N} w_{\text{inv}} = N w_{\text{inv}} \ , \tag{11}$$

where w_{inv} is the computational work required to invert the transport operator.

For the acceleration methods we will consider, and most acceleration methods we have seen in practice, the fundamental process of the sweep is not changed. Additional work is done somewhere in the algorithm and the goal is that the work per sweep w'_{inv} or the number of sweeps N' (or both) decreases. Thus, a modified method is accelerated if

$$\begin{aligned} W > W' \implies & N w_{\text{inv}} > N' w'_{\text{inv}} \\ \implies & N > N' \text{ and/or } w_{\text{inv}} > w'_{\text{inv}}. \end{aligned} \tag{12}$$

We note that this consideration assumes that the only difference between the accelerated solve and the base case is the acceleration itself and that any associated code changes did not have a meaningful impact.

This method of measure highlights the motivation of this work: that it is difficult to accurately assess improvement. Comparing the number of sweeps is can be a useful measure, but this assumes that $w'_{\text{inv}} = w_{\text{inv}}$. If that equality were true, then how do we account for the work being done by the acceleration method? Only counting sweeps does not provide the full picture. Reporting on runtime then seems the best way to measure work, but getting consistent and accurate measures of runtime is not always easy and may not indicate how well a method will translate to different computer hardware. Further, neither method measures exactly what is being improved and if the improvement is what we expected to happen. We are therefore in search of more complete and meaningful ways to measure the value of an acceleration method and that is what we are aiming for with BART.

CODE DESIGN GOALS

The motivation for the Bay Area Radiation Transport (BART) code is to create a laboratory for the development and assessment of novel acceleration methods. The main design goals of BART are to create a code that

1. relieves some of the burden of implementing a novel acceleration method,
2. provides a controlled environment for measuring the effectiveness of the novel method, and
3. provides tools for verifying the basis for improvement.

The first goal is purely practical; once a method has been mathematically derived, it still must be implemented in a code base to produce numerical results. The second and third goals are informed by the discussion in the previous section, and form the basis for an assessment of the effectiveness of our novel acceleration method. In the following sections, we will discuss each of these goals and how BART is designed to meet them.

Easing implementation

The first design goal of BART is to ease the implementation of novel acceleration methods or combinations of methods. BART is being designed with a developer end-user in mind. The code base is not designed to be optimized for speed, unlike many existing codes developed in academia and at national labs. Instead, it is optimized for modification. We have attempted to design the code base to discretize portions of the transport solve, making it clear where all actions are taking place. This focus on clarity instead of performance in the code is aimed to enable researchers to quickly identify exactly what portions of the code need to be modified to implement a new method. This effort is targeted at nuclear engineers with an understanding of the transport solve process, not computer scientists. Ideally, this approach should reduce the learning curve required to begin implementation of a new method when compared to a more complex code base.

BART leverages a feature of object oriented programming, polymorphism, to simplify development. Sections of the transport solve are defined by a single abstract interface, an object that only defines *what* it needs to do, not *how*. Developers are free to customize portions of the solve without having to modify existing implementation. Their new method merely plugs into the solver where the old one was without needing to modify any other parts of the code. Default implementations of all parts of the solve are included, so users can focus on implementing their novel acceleration method without needing to write an entire transport solver.

Finally, the BART code is designed with high standards for testing the default implementation. Users who develop new methods have the benefit of using a tested framework and can verify their modifications have not had adverse impacts on other portions of the solve.

Assessing effectiveness

The second and third design goals of BART are the true drivers of development. First, we aim to make BART a code that provides a controlled environment for measuring the effectiveness of acceleration methods. As mentioned, polymorphism allows users to develop completely novel methods and plug them in seamlessly. This process leaves the unaffected parts of the transport solve unmodified, resulting in the minimum change to implement the method. This avoids the insertion of logic trees and branch-off points that bypass old methods to institute new ones. We are then empowered to compare our base case to our accelerated case with our changes isolated from impacting other portions of the process.

The second part of measuring effectiveness is verifying that the acceleration method requires less computational work. As previously discussed, this is not always straightforward and the situation is especially complicated when combinations of methods are applied. With BART, we aim to give this researcher enough data to make an informed assessment and present their justification. The data that would help inform this assessment is not always easily available in existing codes. Collecting *in situ* data during a solve is rarely efficient for codes designed to focus only on convergence, unless that

data is necessary for the solve to complete. BART is designed with a focus on instrumenting the solve, enabling the user to collect various data throughout the convergence process. This data collection provides information about the overall effectiveness of acceleration methods, and the effectiveness at various iterations in the process.

The focus on instrumentation also works to fulfill the third design goal of BART, to enable users to verify the basis for improvement of their method. Ease of data collection during the solve will help researchers determine if their method is effective for the reasons they hypothesize. One example of instrumentation to this end is Fourier analysis. Acceleration methods are often designed to reduce persistent error modes with less computational work than the base scheme. BART is designed with this analysis in mind, with a native ability to store the solution at each iteration of the solve. Often, this is not practical given the size of some problems. To resolve this issue, BART is planned to have a built-in fast Fourier transform instrument that will calculate the modes of the error at each iteration and store only their amplitudes. Although this requires the code to be run twice, it has the benefit of not needing much storage space or to conduct follow-on Fourier analysis.

THE BART CODE

BART is coded in C++ and uses many features only codified in the C++17 standard. Using this modern C++ standard provides access to better containers and more efficient algorithms as well as various improvements that make the code easier to read and modify. Unit testing and mocking are implemented using the Google Test library [4]. Online continuous integration and code coverage checking are used to ensure the source builds properly and to minimize portions of the code not covered by unit tests. Robust testing ensures that results are reliable and reproducible. In addition, the use of polymorphism and the minimization of code changes need to implement new methods enhances portability. Two BART users can exchange single classes and insert them into their own solves.

BART is a deterministic finite-element-based transport solver that supports one-, two-, and three-dimension solves. The `deal.II` finite-element library [5] provides the underlying calculation infrastructure. This dependency provides meshing, finite-element function values, and integration with PETSc [6, 7, 8] for parallel processor (MPI) calculations.

The code is designed with second-order formulations of the transport equation in mind. The initial version in development has default implementationss of the self-adjoint angular flux equation [9] and diffusion equation. This version will also include source and power iteration, and implement two acceleration methods: nonlinear diffusion acceleration [10], and the transport two-grid acceleration scheme [11]. A Python interface is planned to enable users to run and automate transport calculations using a Python script. This will prevent the use of error-prone text input files.

CONCLUSION

In this work, we have discussed iterative methods and the way we assess the effectiveness of acceleration schemes meant to improve convergence. We have described a new code, BART, designed to help researchers and developers assess the effectiveness of these schemes by providing enough data to assess the effectiveness of methods, enabling deeper understanding than what is available from runtime and sweep count alone. By minimizing and isolating the modifications needed to implement new methods, the code is also designed to make results and methods portable and reproducible. We hope that this code will ease the practical challenges involved in assessing the effectiveness of methods. In doing so, we hope to advance the state of the art in acceleration methods.

REFERENCES

1. Warren F. Miller and Elmer E. Lewis. *Computational Methods of Neutron Transport*. American Nuclear Society, 1993.
2. Marvin L. Adams and Edward W. Larsen. Fast iterative methods for discrete-ordinates particle transport calculations. *Progress in Nuclear Energy*, 2002.
3. Brian Nease, Forrest Brown, and Taro Ueki. Dominance ratio calculations with MCNP. *International Conference on the Physics of Reactors 2008, PHYSOR 08*, 1(505):708–714, 2008.
4. Google. Googletest – Google Testing and Mocking Framework. `https://github.com/google/googletest`. Accessed: 2017 December 20.
5. et al. G. Alzetta. The `deal.II` library, version 9.0. *Journal of Numerical Mathematics*, 26(4):173–183, 2018.
6. et al. Satish Balay. PETSc Web page. `https://www.mcs.anl.gov/petsc`, 2019.
7. et al. Satish Balay. PETSc users manual. Technical Report ANL-95/11 - Revision 3.12, Argonne National Laboratory, 2019.
8. Satish Balay, William D. Gropp, Lois Curfman McInnes, and Barry F. Smith. Efficient management of parallelism in object oriented numerical software libraries. In E. Arge, A. M. Bruaset, and H. P. Langtangen, editors, *Modern Software Tools in Scientific Computing*, pages 163–202. Birkhäuser Press, 1997.
9. J E Morel and J M Mcghee. A Self-Adjoint Angular Flux Equation. *Nuclear Science and Engineering*, 132:312–325, 1999.
10. H. Park, D. A. Knoll, and C. K. Newman. Nonlinear Acceleration of Transport Criticality Problems. *Nuclear Science and Engineering*, 172(1):52–65, 2012.
11. Thomas M. Evans, Kevin T. Clarno, and Jim E. Morel. A Transport Acceleration Scheme for Multigroup Discrete Ordinates with Upscattering. *Nuclear Science and Engineering*, 165:292–304, 2010.